Personalizing Computer Science Education by Leveraging Multimodal Learning Analytics

David Azcona
Insight Centre for Data Analytics
Dublin City University
Dublin, Ireland
David.Azcona@insight-centre.org

I-Han Hsiao
School of Computing, Informatics & Decision Systems Engineering
Arizona State University
Tempe, Arizona, USA
Sharon.Hsiao@asu.edu

Alan F. Smeaton
Insight Centre for Data Analytics
Dublin City University
Dublin, Ireland
Alan.Smeaton@insight-centre.org

***Abstract*—This Research Full Paper implements a framework that harness sources of programming learning analytics on three computer programming courses a Higher Education Institution. The platform, called PredictCS, automatically detects lower-performing or "at-risk" students in programming courses and automatically and adaptively sends them feedback. This system has been progressively adopted at the classroom level to improve personalized learning. A visual analytics dashboard is developed and accessible to Faculty. This contains information about the models deployed and insights extracted from student's data. By leveraging historical student data we built predictive models using student characteristics, prior academic history, logged interactions between students and online resources, and students' progress in programming laboratory work. Predictions were generated every week during the semester's classes. In addition, during the second half of the semester, students who opted-in received pseudo real-time personalised feedback. Notifications were personalised based on students' predicted performance on the course and included a programming suggestion from a top-student in the class if any programs submitted had failed to meet the specified criteria. As a result, this helped students who corrected their programs to learn more and reduced the gap between lower and higher-performing students.**

***Index Terms*—Computer Science Education, Learning Analytics, Predictive Modelling, Peer Learning, Machine Learning, Educational Data Mining**

I. Introduction

PredictCS is a Predictive Analytics platform for Computer Science courses that notifies students based on their performance using past student data and recommends most suitable resources for students to consult [1]. The first implementation of this framework was on an introductory computer programming course in 2017 [2]. This implementation trained a model of student performance using one year of groundtruth with data features including engagement and programming effort. Pseudo real-time predictions were run on a new cohort of students while "at-risk" students were targeted during laboratory sessions.

The framework has since been updated by the implementation of a new multimodal predictive analytics system that aggregates sources of student digital footprints from blended classroom settings [3]. Advanced data mining techniques are adopted to engineer models to provide realtime prediction and dynamic feedback. Preliminary results on a programming course show the potential of this solution. In this work, the system is implemented in three more computer programming courses. The models add further features, namely static student information along with more engagement features. In short, the system uses machine learning techniques and predicts student performance in three programming courses at first and second-year undergraduate level.

A retrospective analysis was carried out to verify the viability of the models, pseudo real-time predictions were run weekly during the teaching period on new cohorts of students, and automatic feedback was sent to students who opted in during the second half of the semester. This feedback was built using our performance predictions and code suggestions from top-ranked students in the class. We propose the following research questions:

RQ1: How accurate are the proposed predictive models with generic static and dynamic student data features identifying students in need in programming courses across a variety of first and second-year programming courses?

RQ2: What are the effects of timely automatic adaptive support and peer-programming feedback on students performances across these courses and what are the differences among them?

RQ3: What are the students and teachers perspectives and experiences on these courses?

II. Related work

In CSEd research, based on the programming events' granularity (type and frequency), different models have been developed for modelling student programming learning behaviour. Researchers have leveraged key strokes, program edits, compilations, executions and submissions [4]. In our university's automated grading system for the teaching of computer programming we collect programming submissions and web logs. We have a fine-grained footprint about each submission but we are limited by the frequency of the students submitting their solutions and we miss the programming actions in-between.

Feedback is an important research avenue for the teaching of programming and an effective way to motivate novice programmers. Recently, researchers have been working on

augmenting the IDE or programming environment by crowd-sourcing code solutions. Students are suggested error corrections or solutions that peers have applied before. Java has been the programming language targeted the most with solutions such as BlueFix [5] or HelpMeOut [6]. This latter social recommender system was also applied to Arduino. In addition, Crowd::Debug [7] was presented as a similar solution for Ruby, a test-driven development language. In terms of notifying students how they are progressing throughout a semester, Purdue University's Course Signals [8] sends a personalised mail and posts a traffic signal as an indicator of their performance while Dublin City University's PredictED [9] project notifies students how they are doing and where they are within their own class. Both systems yielded impressive improvement in first-year retention rates. Our programming grading system also provides real-time feedback on each submission by running a suite of test cases but provides no code suggestions or personalised help for errors.

Researches have focused on generating models which accurately predict student performance and identifying which factors carry more predictive power. However, after identifying those "at-risk" students, we should intervene and help those students. Targeting these weak students during laboratory sessions can aid some students [2], but lecturers usually do not have the time or resources to support many students in large classes or to spend as much time identifying what the student knows and does not know. Automatic interventions for programming classes are having great success in other institutions and environments and we are eager to develop our own strategies using our platforms and resources.

III. Data Collection

Dublin City University's academic year is divided in two semesters with a week of inter-semester break in between. Semesters are comprised of a 12-week teaching or classes period, 2-week study period and 2-week exam period. Laboratory sessions and computer-based examinations are carried out during the teaching period. Our previous study [2] was done on Computer Programming I, CS1, that introduces first-year students to computer programming and the fundamentals of computational problem solving during their first semester. We work with the following courses that are taught in the second semester (Fall):

- **Computer Programming II, CS2**: This course introduces first-year students to more advanced programming concepts, particularly object-oriented programming, programming libraries, data structures and file handling. Students are expected to engage extensively in hands-on programming with the Python programming language. A previous version of CS2 was taught using Java before it was redesigned. The current version with Python has been taught for two academic years, 2015/2016 and 2016/2017. The course is a continuation of Computer Programming I, an introductory programming course.
- **Managing Enterprise Computer Systems, SH1**: This course equips first-year students with the basic skills necessary to administer modern enterprise operating systems and shows students how to manage Unix and Unix-like systems. Specifically, they study the Unix shell and work shell scripting programming exercises using tools like test, find or grep and concepts like loops, pipes or file handling. This course has been taught for the past seven academic years since 2010/2011. Students work with the Bash Unix shell and the command language.
- **Programming Fundamentals III, PF3**: This course teaches second-year students fundamental data structures and algorithms in computational problem solving. The material includes linked lists, stacks, queues or binary-search trees; and other techniques, like recursion. This is a new course that has been taught for the first time this academic year 2016/2017 and the language chosen is Python. PF3 is a continuing course of **Programming Fundamentals II, PF2**, that was taught to second-year EC students for the first time on the first semester. Even, Programming Fundamentals I, PF1, was taught to first-year students for the first time this year but second-year students could not take it last year as it didn't exist. In PF2, students learn to design simple algorithms using structured data types like lists and dictionaries, and write and debug computer programs requiring these data structures in Python. This course is also taught in Python. PF3 uses the Python language. It emerged along with PF1 and PF2 for a need for enterprise computing students to have deeper computer programming skills in the workplace.

In all courses, students are assessed by taking two laboratory computer-based programming exams, a mid-semester and an end-of-semester assessment, during the teaching period. In PF3, instead of an end-of-semester lab exam, students demo a project. Each laboratory exam or demo contributes equally to their continuous assessment mark; 15% in CS2, 25% in SH1 and 20% in PF3. Students are not required to submit their laboratory work for SH1 or PF2. In contrast, laboratory work count towards their final grade of the course for CS2 and PF3, both 10% of the overall grade for the course. The CS1 Lecturer developed a custom Virtual Learning Environment (VLE) for the teaching of computer programming. This automated grading platform is currently used in a variety of programming courses across CS; including CS1, CS2, SH1, PF2 and PF3. Students can browse course material, submit and verify their laboratory work. This platform has been used for the past two academic years and that is the data we are using for our analysis.

In terms of numbers, 134 students registered for CS2 during 2015/2016 and 140 students in 2016/2017. For SH1, 70 and 81 students enrolled respectively for the past two academic years. For PF3, 60 students registered in 2016/2017 as that has been the only version of the course. Enrolment numbers from previous years are shown in Figure 1. CS1 and PF2 are taught in the first semester of the academic year. CS2, SH1 and PF3, the main subject of this study, are taught during the

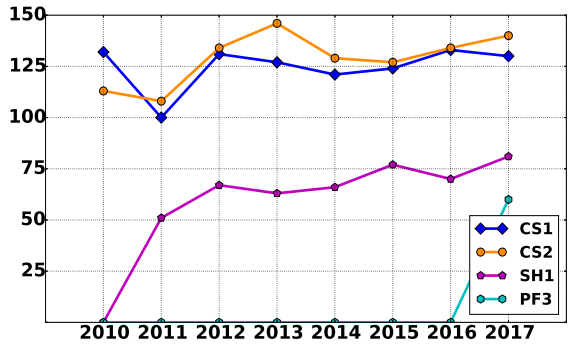

Fig. 1. Enrolment numbers on the courses studied

second.

IV. Predictive Modelling

Predictive Analytics models were developed to automatically classify students having issues with programming material. For each of the courses introduced a distinctive predictive model was built, one per course. These models use the student's digital footprints to predict their performance in computer-based laboratory programming assessments.

A. Student's Digital Footprint

For each student, we build a digital footprint by leveraging the data modalities available and modelling student interaction, engagement and effort in programming courses. The data sources are the following:

- Student Characteristics
- Prior Academic history
- Programming submissions
- Behavioural logs

Features are handcrafted using those data sources. In our previous work [10], we analysed 950 first-year Computer Science (CS) entrants across a seven year period and showed the significant relationship between their entry points or math skills and how they perform in their first year and in programming courses at our university.

B. Training predictive models

A classification model was built for each course to distinguish "at-risk" students. CS2 and SH1's models were trained with historical student data from 2015/2016. In contrast, we did not have any student training data for PF3 as 2016/2017 was the first academic year that course had being taught. As a workaround, we trained a model using Programming Fundamentals II (PF2)'s 2016/2017 student data which was taught during the first semester. The target was to predict whether each student would pass or fail their next laboratory exam. These programming courses are quite dynamic and programming laboratory exercises vary considerably from year to year. However, concepts and knowledge being taught should remain the same.

TABLE I
Course pass rates on groundtruth data

Course	Year	Semester	# Students	1-Exam	2-Exam
CS2	2015/16	2	149	44.30%	46.98%
SH1	2015/16	2	73	32.88%	69.33%
PF2	2016/17	1	60	51.67%	36.67%

After deriving the features, a model for each course was trained to predict a student's likelihood of passing or failing the next computer-based laboratory exam. We developed classifiers for each week of the semester. In 2015/2016, there was a mid-semester exam and an end-of-semester exam for the courses we are building our groundtruth with: CS2, SH1, PF2. To clarify, classifiers from week 1 to 6 were trained to predict the mid-semester's laboratory exam's outcome (pass or fail for each student) and from 7 to 12 the end-of-semester's laboratory exam's outcome.
The Empirical Error Minimization (EMR) approach has been employed to determine the learning algorithm with the fewest empirical errors from a bag of classifiers C [11, 2]. On these models, instead of taking the learning algorithm with the lowest empirical risk or the highest metric (namely accuracy or F1-score), we looked at these metrics per class (pass or fail). Generally, the results on the next laboratory exam, our target variable, are quite imbalanced as in some courses there might many more students that pass rather than fail an exam. The resulting accuracy of a learning algorithm could be misinterpreted if we weight the predictions based on the numbers per class. Our goal is to identify weak students as we prefer to classify students "on the edge" as likely to fail rather than not flagging them at all and miss the opportunity to intervene and help. See the training data pass rates in Table I.

C. Retrospective Analysis

Following a customized EMR approach, we selected a classifier for each course that minimized the empirical risk on average for the 12 weeks looking at the fail class and had a good balance between both classes. The learning algorithms and their corresponding metric values on our training data are shown in Table II. For instance, in CS2, we selected a K-Neighbors classifier which gave us a high F1-metric on average and the highest on weeks 5 and 6. Those weeks are key to identify who is struggling before their first assessment. Figure 2 shows the performance of the bag of classifiers for each week on average of the Cross-Validation folds. Figure 3 only shows the likely-to-fail class. Following this approach, in CS2, we selected a K-Neighbors classifier which was the second-highest on average for the F1-metric looking at both classes. The highest was a Random Forest which was giving the values 75.39% and 62.25% for the F1-score metric for the fail and pass classes respectively. However, two classifiers on week 5 and 6 were getting slightly better results by the K-Neighbors classifier and those weeks are key to identify who is struggling. We decided to choose the K-Neighbors then so we could better identify them before their first laboratory

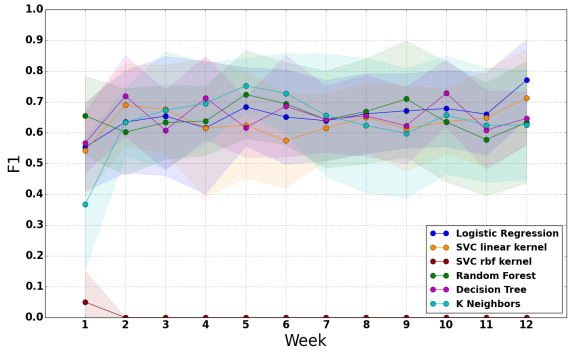

Fig. 2. CS2 F1 Classifiers Performance

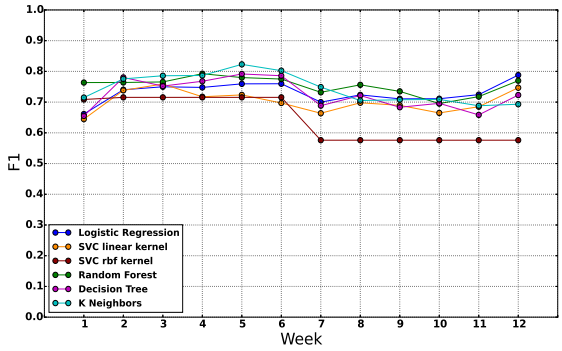

Fig. 3. Performance of the bag of classifiers using the F1-metric for course CS2 looking at the fail class

examination on week 6. In SH1, Random Forest looked like the most promising classifier with the highest fail class F1-score, 74.26% but a very low value for the pass class: 32.20%. The pass class values were particularly very low the first few weeks of the semester as it was classifying most of the students as failing the next laboratory exam which does not help us that much. We went for an SVM with linear kernel with the values shown in Table II. See Figures 4 and 5. We can see how the SVM with Gaussian kernel fails to learn and predicts all students to fail. It gets good accuracy some weeks of the semester but we are aiming for a good balance for both classes with an emphasis on the failing class. Lastly, for PF2, we picked up the Decision Tree classifier as it gave us the highest F1-score for the fail class and one of the highest for the pass class. Recall, PF3's model is based on PF2's training data. See Figures 6 and 7.

In addition we ran a statistical significance for the classifiers selected with respect to the others in the bag of classifiers. The predictions were only statistically significant compared to to the SVM with Gaussian kernel which failed to learn properly. This indicates the predictions are very similar as we only have groundtruth data for one academic year so far and we do not have enough information to determine the selected classifiers' predictions were statistically independent.

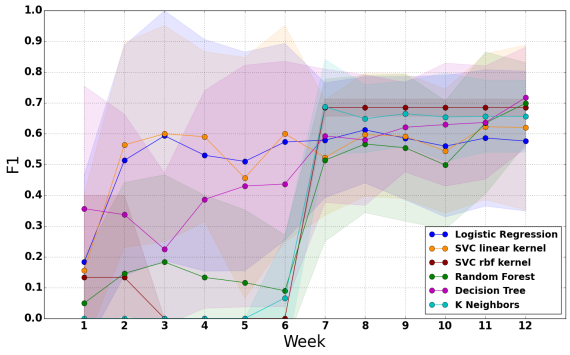

Fig. 4. SH1 F1 Classifiers Performance

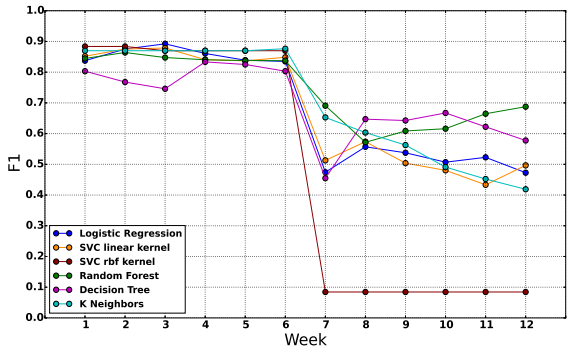

Fig. 5. SH1 F1 Fail class Classifiers Performance

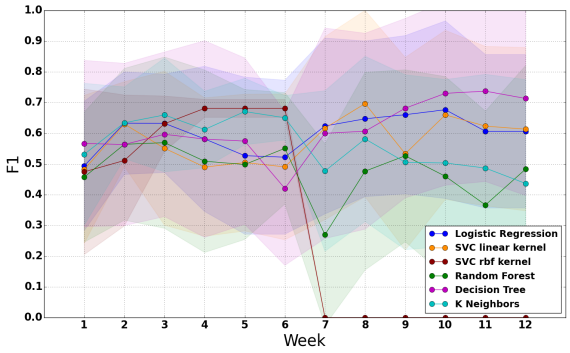

Fig. 6. PF2 F1 Classifiers Performance

TABLE II
LEARNING ALGORITHM SELECTED FOR EACH COURSE

Course	Algorithm	Class	F1-score	Precision	Recall
CS2	K-Neighbors	Fail	74.50%	71.41%	81.03%
		Pass	59.81%	68.80%	58.74%
SH1	Linear SVM	Fail	67.77%	71.46%	69.54%
		Pass	41.61%	45.53%	42.82%
PF2	Decision Tree	Fail	66.73%	68.58%	72.14%
		Pass	55.00%	58.62%	59.64%

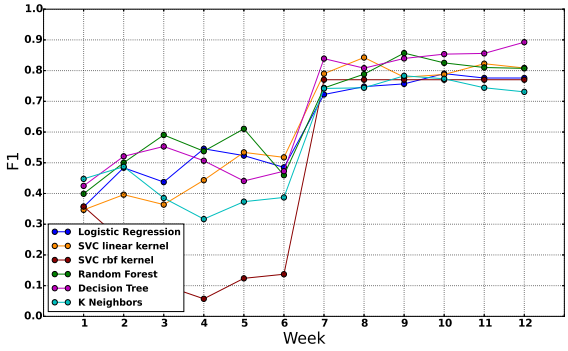

Fig. 7. PF2 F1 Classifiers Performance for the likely-to-fail class

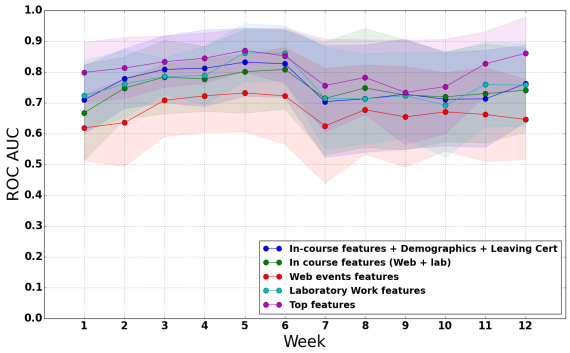

Fig. 8. CS2 ROC AUC Features Analysis

TABLE III
CORRELATION BETWEEN THE FEATURE VALUES AND THE TARGET PERFORMANCE GRADES TO BE PREDICTED

Course	Feature description	Pearson	Spearman
CS2	CS1 Exam-2 2015/2016	35%*	40%*
CS2	Programming Week 6	73%*	73%*
SH1	Years on this course	-31%*	-26%
SH1	Week(end) rate Week 10	36%*	35%*
SH1	Programming Week 12	51%*	51%*
PF2	Hours Spent Week 8	48%*	45%*

* $p - value < 0.01$

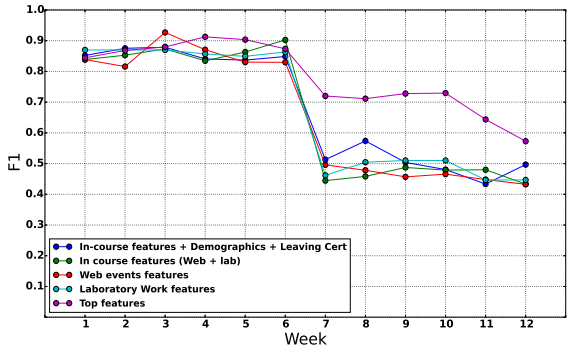

Fig. 9. Fail class Feature Analysis using the F1-metric for course SH1 looking at the fail class

We measured the predictive power of our features by calculating the correlation between the students' grades and our target, the next laboratory exam results, using the linear (Pearson) and non-linear (Spearman) correlation coefficients. Table III shows examples of previous academic performance, static characteristics, interaction and programming features. This analysis confirms the power of our features and the programming weekly and cumulative progress features increasingly gain importance throughout the semester as students put more effort into the courses.

In a similar manner, we built a trees classifier for each course and per week that fits a number of randomized decision trees. By building this type of forest, we were able to compute the importance for each feature. Based on this forest of importances, we selected the top 10 features every week to avoid over-fitting. A similar comparative analysis to the classifier analysis was run to verify this approach for feature selection improved our metric values, see Figure 8 and Figure 9 for CS2 and SH1 respectively [12]. In addition, Figure 10 shows the top features for a particular week for CS2.

This retrospective analysis shows us we can successfully gather student data about their learning progress in those programming courses leveraging students' digital footprints.

V. STUDENT FEEDBACK

Students that decided to opt-in received weekly customized notifications via email. After their first laboratory exam in week 6, a feature was enabled in the grading platform platform for the teaching of computing programming where students could freely opt-in or out from these notifications and read about the project. After it was enabled, students were not able to submit any programs before they either opted-in or out. Table IV shows the number of people who replied to the opt-in option. Most students opted-in on each course and they received weekly notifications from that moment onwards.

The customized notifications were personalized by leveraging our weekly predictions. Based on the associated probability

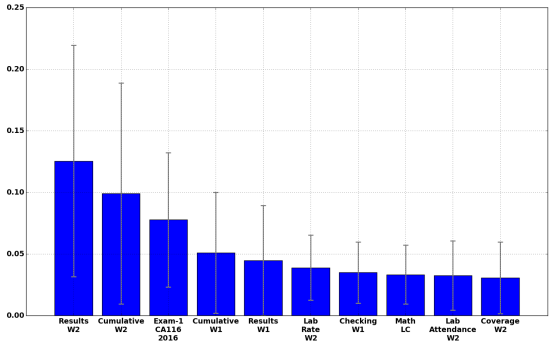

Fig. 10. CS2 Top Features Week 2

TABLE IV
STUDENT OPT-INS AND OPT-OUTS TO INTERVENTIONS

Course	Replied	Opt-ins	Opt-outs	No-reply
CS2 (140)	122 (87.14%)	111 (90.98%)	11 (9.02%)	18 (12.86%)
SH1 (81)	70 (86.42%)	63 (90.00%)	7 (10.00%)	11 (13.58%)
PF3 (60)	51 (85%)	42 (82.35%)	9 (17.65%)	9 (15.00%)

of failing the next laboratory exam, we ranked students, and divided them in deciles. Hence, there were 10 custom messages we sent based on their performance. In addition, for each notification, we included one programming suggestion if the student had submitted a program that failed any of the testcases and had been considered incorrect. We utilized a matching algorithm to suggest the closest program from a correct solution developed by a top-performer in the class that week. The top students are the 10% highest ranked in that class from our predictions each week. We recommended the closest submission by text similarity between the programs after removing the comments. At the end of the note, students could be redirected to read the Terms or unsubscribe from these notifications if desired. Nobody unsubscribed from these notifications throughout the semester. See Figure 11 for a sample of the notifications.

VI. RESULTS

Predictions are run on data from 2016/2017's incoming student cohorts. We analyse the results obtained by running predictions along with the feedback sent to them and what this means for the research questions proposed.

A. RQ1: Predictions

Predictions were run on a pseudo real-time basis every week for students who registered for the three courses during the second semester of 2016/2017 using the models trained with our groundtruth data. Individual reports were emailed to the lecturers every week and posted on our analytics web application accessible to them at any time. In order to evaluate how our predictions performed, we compared the corresponding weeks' predictions with the actual results of the two laboratory exams that took place in weeks 6 and 12 in 2016/2017 for each course. Table V contains details of the accuracy of our predictions. Overall, these worked quite well. As the semester progressed, our early alert system gathered more information about students' progression and our classifiers were able to learn more as shown by the increased accuracy and F1-score measures and the decreasing number of students flagged as "at-risk". In short, we could automatically distinguish in a better way of who is going to pass or fail the next laboratory exam.

B. RQ2: Interventions

Notifications were sent to opt-in students for the second half of the semester classes as shown in Table VI. For instance, in CS2, 438 notifications were sent to students, 181 of these contained a programming suggestion and the remaining did not have any program to suggest to the student. 21 of those programs suggested were corrected by the students after the recommendation was sent.

We extracted and created two groups from the students, the ones that corrected any of the programs suggested to them with the solution outlined or another solution, and the ones who were suggested one or more programming suggestions but did not correct any. This can be found in Table VII. In SH1, there were 16 students who corrected some of their failing programs as suggested and 35 that were suggested but they did not. For the ones that did, their average grade on the second exam was more than 2 points below but for the ones that did not, their's was more than 11 points. A lesser learning improvement is observed for CS2 and PF3. Based on only year of data, students seem more motivated to learn from the programs that are offered to them as advice for failed submissions.

Instead, students can also be grouped into higher and lower-performing groups based on their results in the first assessment in order to measure the impact of our notifications and the gap between them and these results are shown in Table VIII. Students who failed CS2 and SH1's first laboratory exam improved almost 2 and almost 5 more points on average respectively on 2016/17 when the predictions were run, students were ranked, programs were suggested and notifications were sent to students. There is no basis for a similar analysis of PF2 and PF3 as those are different courses from the same academic year and the student cohort was the same.

In both scenarios, students who learned from the programs suggested and lower-performing students, showed a learning improvement over the two other groups.

C. RQ3: Students have their say

In any student intervention, it is important to get students' opinions about the feedback introduced to them, understand how it affects their behaviour within the courses and if it encourages them to try new solutions or to revise material. Thus, we gathered their opinions about our system and the notifications via a written questionnaire. The questions on the form were the following:

Q1: Did you opt-in? Yes / No
Q2: If you opted-out, could you tell us why?
Q3: How useful did you find the weekly notifications . . . 1 to 5 rating
Q4: Did you run any of the suggested working programs? Yes / No / I was never suggested any
Q5: Would you recommend the system to a student taking this same course next year? Yes / No
Q6: Would you like to see weekly the system notifications for other courses? Yes / No
Q7: How could we improve the system for next year? Any other comments.

In CS2 and SH1, the questionnaire was filled the second-last day of the semester classes during an evaluation for another course where all CA and EC first-year students should have attended. That allowed us to gathered a good amount of

SH1 PredictCS Weekly Feedback

Dear Jane Doe,
For the last week our records show that you are engaging well with the courseware and seem to be managing module SH1 well. Well done you. Please keep up the good work this week for the module. Lab attendance is generally correlated with the student's performance. Well done to you for attending last week's lab session.

In addition, your last submission for **run-if-safe.sh** on labsheet-06 was:

```
if sha256sum -c checksum.txt | grep OK | cut
then
  sh suspicious.sh
else
  false
fi
```

Check out and give a try to this working version of **run-if-safe.sh** that by design is the closest to yours in the class:

```
if sha256sum -c checksum.txt | grep -q -w OK
then
   sh suspicious.sh
else
   false
fi
```

Notice
Our predictions and recommended programs are based on your engagement and effort with the course (the programs you develop and the course material you access); your characteristics and prior performance. Remember, this is just our best guess as to how you have been doing and is not an indicator of how you will do in the module, laboratory exams or written exam. However, if you need to improve, it is easy to make a change for next week; just spend more time programming and come to the labs, submit your programs to Einstein or access the material on non-lab days. Please use this information to help you to increase your motivation and engagement with SH1. Contact us at predictcs@university.edu for further details.

If you feel as though you need additional supports to help you with this, please contact your lecturer Peter Smith at peter.smith@university.edu or Student Support & Services at student.support@university.edu.

Terms | Opt Out

Fig. 11. Personalized notification sent to a student in SH1

TABLE V
COURSE INFORMATION, PASS RATES, AT-RISK PREDICTION RATES, PREDICTION METRIC RESULTS AND CORRELATIONS PER COURSE

Course	Year	Sem.	Students	Exam	Passing	At-risk	Accuracy	Precision	Recall	F1-score	Pearson	Spearman
CS2	2016/17	2	133	W6	58.57%	74.29%	64.29%	94.44%	41.46%	57.63%	0.62 (0.0000)	0.60 (0.0000)
				W12	42.86%	44.29%	77.14%	67.95%	88.33%	76.81%	0.65 (0.0000)	0.70 (0.0000)
SH1	2016/17	2	81	W6	70.37%	80.25%	46.91%	93.75%	26.32%	41.10%	0.41 (0.0001)	0.40 (0.0002)
				W12	69.14%	48.15%	67.90%	85.71%	64.29%	73.47%	0.41 (0.0002)	0.42 (0.0001)
PF3	2016/17	2	60	W6	88.33%	48.33%	56.67%	93.55%	54.72%	69.05%	0.42 (0.0008)	0.45 (0.0004)
				W12	90.00%	65.00%	45.00%	100%	38.89%	56.00%	0.43 (0.0006)	0.45 (0.0003)

TABLE VI
INTERVENTIONS

Course	Notifications	Suggestions	# Corrected	% Corrected
CS2	438	181	21	11.60%
SH1	238	134	25	18.66%
PF3	165	80	8	10.00%

TABLE VII
LEARNING IMPROVEMENT

Course	Class	Num	1-Exam	2-Exam	Impr.	Diff.
CS2	Yes	16	32.81%	27.62%	-5.19%	+5.85%
	No	53	45.28%	34.24%	-11.04%	
SH1	Yes	16	60.94%	58.75%	-2.18%	+8.96%
	No	35	58.57%	47.43%	-11.14%	
PF3	Yes	7	53.57%	73%	+19.43%	+3.74%
	No	28	58.25%	73.92%	+15.68%	

responses for both courses. See the first column on the table which is the response rate. In PF3, the questionnaire was filled

TABLE VIII
IMPROVEMENT BETWEEN HIGHER AND LOWER-PERFORMING STUDENTS OVER THE TWO ACADEMIC YEARS

Course	Year	Cohort	1-Exam	2-Exam	Improvement.	Differential	Learning differential
CS2	2015/16	Passed 1st Exam	75.23%	55.06%	-20.17%	+29.87%	+1.89%
		Failed 1st Exam	14.70%	24.40%	+9.70%		
	2016/17	Passed 1st Exam	76.22%	47.85%	-28.37%	+31.76%	
		Failed 1st Exam	8.62%	12.02%	+3.40%		
SH1	2015/16	Passed 1st Exam	64.17%	64.17%	+0.00%	+35.29%	+4.89%
		Failed 1st Exam	5.88%	41.18%	+35.29%		
	2016/17	Passed 1st Exam	72.81%	55.44%	-17.37%	+40.18%	
		Failed 1st Exam	17.19%	40.00%	+22.81%		

TABLE IX
SURVEY RESPONSES FROM STUDENTS ABOUT THE PROJECT

Course	Response	Q1	Q3	Q4	Q5	Q6
CS2	75.71%	93.40%	3.49 ⋆	33.70%	85.15%	83%
SH1	80.25%	84.60%	3.82 ⋆	40.40%	91.33%	91.38%
PF3	53.33%	87.50%	3.45 ⋆	21.88%	100%	90.00%

during the laboratory session in week 11.
Overall, feedback was very positive and responses can be found in Table IX. Most students would recommend this system to students attending the same course next year or would like to see this system included in other courses as shown in questions 5 and 6 respectively. In terms of the last question to improve the system, students who were doing well or very well, were getting an increasingly similar response each week and were demanding a more personalised notification and some other additional learning resources.

VII. DISCUSSION

Predictions worked relatively well with one year of training data for the three courses. Next year we will be able to generalise far better by having two different student cohorts from which to extract our usage patterns. We should note again, CS2 and SH1's models were based on 2015/16's previous student data and PF3's was based on PF2's student data from the first semester of the academic year. We could not expect it to work as well as the other models as the courseware is not the same, the concepts taught are similar but more advanced and utilizing PF2's patterns has surprisingly resulted in good outcomes. Hence, we are aplying now models for new courses based on our whole student dataset from all courses available. The more data is usually the better and even though the material and programming exercises are different, we might be able to identify at-risk students using general engagement, progress and static patterns. In addition, we believe CS2's actual results were more correlated with our predictions than SH1's as CS2 is an advanced course that requires more studying time from students. Both are hands-on programming courses but CS2 is designed for prospective Software Engineers in CA and the concepts taught are harder programming wise. The progression and effort students need to invest in seems higher and, hence, our predictors work better in identifying who is at-risk in CS2. In terms of the notifications, after enabling the feature to opt-in or out on the courses' courses, students had to log in to the submission platform to select either option. Most of the students that replied did that in the first or second week, specially on the laboratory sessions where the use the platform to verify their work. A few "tardy" students did it later and did not receive their previous notifications even if they had opted-in after. Others were completely disengaged and never got to reply. The approach we chose for the programming recommendations was to pick the closest text program from top-ranked students in the class that year. This could be further advanced by identifying variables and choosing the closest program syntactically and semantically. Other approaches we tested before were Collaborative Filtering as recommender systems use today by looking at the closest person to you in the class or within the top-students, and to recommend one of their programs. Netflix recommends movies or Amazon recommends products from people with the same tastes assuming they are constant, we could assume programming design is constant and recommend the students program from the closest person. We were more interested in the students being able to identify what's wrong with their problems and the closest solution from a top-student worked very well, especially for shorter programs like the ones suggested in SH1. We could also use crowdsourcing and recommend the program uploaded the most by using previous and current year solutions. Even lecturers were also providing some sample solution and explanatory code after seeing the students' reactions to the notifications in those three courses.

VIII. CONCLUSION & FUTURE WORK

We are extending the implementation of this framework to more programming courses. In addition, we are being more vocal about the project so more students can benefit from this research methodologies. The following years, we will generalize and model the students' behaviour on these courses better as we will be able to train the models with several historical student cohorts. We are excited to personalized, enhace and motivate our learners by providing them with more detailed programming recommendations, suitable resources and other actions to fill the knowledge programming holes they may have while learning CS programming design at our university.

ACKNOWLEDGEMENTS

This research was supported by the Irish Research Council in association with the National Forum for the Enhancement of Teaching and Learning in Ireland under project number GOIPG/2015/3497, by Science Foundation Ireland under grant number 12/RC/2289, and by Fulbright Ireland.

REFERENCES

[1] David Azcona, I-Han Hsiao, and Alan Smeaton. "PredictCS: Personalizing Programming learning by leveraging learning analytics". In: (2018).

[2] David Azcona and Alan F Smeaton. "Targeting At-risk Students Using Engagement and Effort Predictors in an Introductory Computer Programming Course". In: *European Conference on Technology Enhanced Learning*. Springer. 2017, pp. 361–366.

[3] David Azcona, I-Han Hsiao, and Alan Smeaton. "Detecting Students-In-Need in Programming Classes with Multimodal Learning Analytics".

[4] Petri Ihantola et al. "Educational data mining and learning analytics in programming: Literature review and case studies". In: *Proceedings of the 2015 ITiCSE on Working Group Reports*. NY, USA: ACM, 2015, pp. 41–63.

[5] Christopher Watson, Frederick WB Li, and Jamie L Godwin. "Bluefix: Using crowd-sourced feedback to support programming students in error diagnosis and repair". In: *International Conference on Web-Based Learning*. Washington, USA: Springer, 2012, pp. 228–239.

[6] Bjorn Hartmann et al. "What would other programmers do: suggesting solutions to error messages". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. NY, USA: ACM, 2010, pp. 1019–1028.

[7] Dhawal Mujumdar et al. "Crowdsourcing suggestions to programming problems for dynamic web development languages". In: *CHI'11 Extended Abstracts on Human Factors in Computing Systems*. NY, USA: ACM, 2011, pp. 1525–1530.

[8] Kimberly E Arnold and Matthew D Pistilli. "Course signals at Purdue: Using learning analytics to increase student success". In: *Proceedings of the 2nd international conference on learning analytics and knowledge*. NY, USA: ACM, 2012, pp. 267–270.

[9] Owen Corrigan et al. "Using Educational Analytics to Improve Test Performance". In: *Design for Teaching and Learning in a Networked World*. Washington, USA: Springer, 2015, pp. 42–55.

[10] "Innovative learning analytics research at a data-driven HEI". In: (2017).

[11] L Gyorfi, L Devroye, and G Lugosi. *A probabilistic theory of pattern recognition*. Washington, USA, 1996.

[12] James A Hanley and Barbara J McNeil. "The meaning and use of the area under a receiver operating characteristic (ROC) curve." In: *Radiology* 143.1 (1982), pp. 29–36.